

## REMARKS

Claims 1-17 are pending in the patent application.

### The Claimed Invention

Claim 1 recites a mobile terminal comprising an operating system having a file system for detecting a filename with an illegal character, wherein the file system comprises an encoder module for encoding the filename by replacing the illegal character with a specific code character having information coded therein about the illegal character itself. Claims 15-17 contain similar limitations. The claimed invention provides a simple symmetrical encoding for filenames to replace each illegal character with a respective specific code character.

Typically, filenames use 16 bit Unicode characters so there is 16 bits to encode one illegal character. According to the present invention, the encoder represents the position of the illegal character in the filename with 8 bits ( $2^8 = 256$ ) and maps the illegal character itself using four bits that represent the Unicode character, as described in the patent application on page 3, lines 10-16.

In operation, the encoder identifies the specific code character from other characters by using the four most significant bits (MSB) as an illegal character indicator, e.g. using the Unicode designation "1110 = E". In the Unicode standard, code areas from E000 to F8FF are reserved for private or internal business use, and available for use as such an indicator.

In one embodiment, the specific code character can be placed at the end of the filename, before the commonly used filename extension. For example, this would make an illegal filename "ale?x.jpg" look like "alex .jpg", where the blank space represents the coded character.

One advantage of the present invention is that the solution makes it possible to receive files, containing illegal characters, from an external source connected to the mobile terminal through WAP, UNIX Server or McIntosh and store the files into the file system without corrupting the filenames. The solution also makes it possible to decode the filenames back to the original format, recreating the illegal characters.

Another important advantage of the present invention is the fact that the mobile terminal can be used as a mass storage device for popular Apple McIntosh operating system even though the file system in mobile does not directly support the MacOS operating system.

Consistent with the aforementioned understanding of the claimed invention, and the advantages thereof, it is respectfully submitted that the reasoning in the outstanding Office Action raises some issues that appear to indicate a possible misunderstanding about the claimed invention, in effect, by possibly trying to fit it so as to comply with a standard Unicode character set. The following is an addition explanation of the claimed invention and the workability thereof that should help to clarify any such possible misunderstanding.

For example, the claimed invention relates to and involves an encoding modification technique where 16 bits (or 32/48/64/128 bits etc), which normally are used to store character mapping information, could be utilized in a different way as shown in Figure 2 of the patent application. The reasoning in the outstanding Office Action appears to be assuming that with a word position (8 least significant bits/LSBs in Fig.2), the claimed invention relates to a given character's position in a Unicode character map, although the claimed invention actually relates to a position within the filename wherein the illegal character (e.g. ?, :, \*, <, >, %, | etc.) is located if the filesystem allows such a character in filenames. Since there is only a restricted set of illegal characters only four bits are needed

to mark them (i.e. the claimed invention uses a customized character mapping) as illustrated in Figure 2a and 2b. In the example shown in Figures 2a, 2b, the four most significant bits (MSBs) are used as "illegal character indicator" when set in a given way e.g. '1110' and thus a filesystem knows that all characters within a filename that have "illegal character indicator" MSBs are treated in a special way (i.e. not as normal Unicode characters but as characters informing about illegal characters that have been removed from the filename).

Further, all those characters with "illegal character indicator" are shown as blank spaces by the filesystem (by contrast, in standard Unicode a blank space is 0x0020 but in the claimed invention those illegal characters coded as 0xE... are also shown as blank spaces). Consequently, filenames like "SunEarth .txt", "Sun>Earth.txt", "Sun?Earth.txt", "Sun:Earth.txt" all would look the same "SunEarth .txt" (if multiple illegal characters then there is more blank spaces at the end of the filename, e.g. "Sun<>Earth?.txt" is shown as "SunEarth .txt"). So a user doesn't see any difference between those filenames and hence there may be several files with similar looking names in the same directory (as opposed to Douceur's patent) but in bitlevel there are differences. In comparison, it is respectfully submitted that preventing duplicate filenames is one of the main advantages of Douceur's patent, so there is no teaching, suggestion or motivation to interpret Douceur's patent in a way the reasoning in the outstanding Office Action is suggesting.

As an example, the illegal character '>' in "Sun>Earth.txt" would be coded as '1110011100000011' (0xE703) and shown as a blank space at the end of the filename since 7 is used for '>' (see Figure 2b), and 3 is the position of the character since the numbering starts from 0 (i.e. fourth character in the filename has position 3, where with 8 bits one can

locate illegal characters within filenames having length of 256 characters). If there were more illegal characters those would be coded with a similar syntax and placed at the end of filename in the order of appearance.

In the case of the filename "ale?x.jpg" which would be shown as "alex .jpg" in a filesystem not allowing '?' in filenames, those allowed characters 'a', 'l', 'e' and 'x' are coded in a standard way whereas '?' is removed and a special character having the "illegal character indicator" (e.g. 0xE) and shown as a blank space is coded as 0xE403 to indicate that it should be decoded as a '?' (see conversion table of Figure 2b) and placed as fourth character from the beginning of the filename (index 3) when transferred to a filesystem allowing such characters. In contrast, it is respectfully submitted that Douceur does not teach or suggest to use character bits in a similar way.

Moreover, the aforementioned discussion concentrates on a scheme where illegal characters are shown as blank spaces at the end of a filename. However, the scope of the invention is not intended to be limited to the same. For example, one may replace an illegal character with a specific code character that has 'illegal character indicator' bits (4 MSBs) set in a given way (e.g. 1110). Thus, in such a situation "ale?x.jpg" would be shown as "ale x.jpg" and the character shown as blank space would be coded as 0xE403 in view of Figure 2 (on the other hand, one could have alternative implementations that are more like standard Unicode since the claimed invention doesn't have to include any position information as the character with a character having information of the replaced character is just replaced and one can hence use standard Unicode mappings by just setting 'illegal character indicator' to 4 MSBs of the 16 character bits. So 8 LSBs could be set to indicate the replaced character's position in standard Unicode character map where '?' is 0x003F and just by adding 'illegal character

indicator' one would get 0xE03F. Most users are likely to select this kind of scheme where they would see where replacement has likely happened (of course, blank space may just be blank space and not a replaced character). Also, it is a system setting how those characters with illegal character indicator are shown (blank space, period, underscore etc.).

Finally, it is respectfully submitted pointed out that the reasoning in the Office Action merely relates to the kind of solution where those 8 LSBs are set to indicate the replaced character's position in the Unicode character map.

Based on this understanding of the claimed invention, the rejections raised by the Examiner are respectfully traversed.

#### The Non-Enablement Rejection

Claim 1 recites that the file system comprises an encoder module for encoding the filename by replacing the illegal character with a specific code character having information coded therein about the illegal character itself. Claims 1-17 are rejected as being non-enabling based on the reasoning that the specification does not describe the subject matter thereof.<sup>2</sup>

The non-enabling rejection is respectfully traversed for the following reasons:

It is respectfully submitted that, when the patent application is read as a whole, including that described on page 6, lines 17-21, a person skilled in the art would appreciate that the specification describes the subject matter of the claimed invention. Clearly, the patent application, on page 6, lines 17-21, states that "a single character is replaced by a code character that knows how to restore the original illegal character back." Moreover, in operation

---

<sup>2</sup> Regarding the remarks in the outstanding Office Action, on page 3, lines 6-10, there appears to be a typo re the use of the character "\$" versus --?--.

and consistent with that discussed above, for example, for the filename "Sun>Earth.txt" having the illegal character '>', the specification makes clear that the encoding would be as follows: The illegal character '>' in "Sun>Earth.txt" would be coded as '1110011100000011' (0xE703) and shown as a blank space at the end of the filename since 7 is used for '>' (see Figure 2b) and 3 is the position since numbering starts from 0 (i.e. fourth character in the filename has position 3 and with 8 bits one can locate illegal characters within filenames having length of 256 characters). The code '1110011100000011' is represented by E703 and contains information coded therein about the illegal character itself, including the position thereof in the filename. Based on this understanding, it is also respectfully submitted that a person skilled in the art would appreciate that the specification provides support for the claim language --a code character having information coded therein about the illegal character itself, -- as claimed.

Because of this, it is respectfully requested that the non-enablement rejection be reconsidered and withdrawn.

#### The Indefiniteness Rejection

On pages 2-3 of the Office Action, the reasoning has also rejected the subject matter of claims 2-4 and 6-7 as not being described in the specification or as being indefinite. However, it is respectfully submitted that the reasoning appears to be overlooking pages 6-8 where these features are described in detail in relation to Figure 2. For example, re claim 2 each character has an inherent "position" in the filename, so such a term does not need an antecedent basis per se. For example, re claim 6, the term "the same" would be understood by a person skilled in the art to mean "the specific code character." In spite of this, the undersigned would be

amenable to making an amendment to claim 6 if the Examiner believes that it would expedite the prosecution of the merits.

### The Anticipation Rejection

The "Response to Arguments" section on page 2 of the Office Action provides a reply to the remarks on page 6 of applicant's November 8th response. The undersigned would like to thank the Examiner for this reply.

The following is a response to the technical points raised by the reply, which also points out why Douceur does not teach or suggest the claimed invention.

Claim 1 recites a mobile terminal comprising an operating system having a file system for detecting a filename with an illegal character, wherein the file system comprises an encoder module for encoding the filename by replacing the illegal character with a specific code character having information coded therein about the illegal character itself. Claims 15-17 contain similar limitations.

As stated in applicant's November 8th response, it is respectfully submitted that Douceur et al. (US 7,047,420) discloses using any character (e.g. letter, number or other symbol, etc.) in place of an underscore in a filename. However, Douceur et al.' replacement character does not contain information coded therein about the illegal character itself, including the position of the illegal character in the filename (Claim 2) and/or the illegal character itself (claim 4). For example, in Douceur if a filename has an illegal underscore, and if either a replacement character "1" or a character "B" is used in place of the underscore, then in either case if the filename is later decoded the character "1" or the character "B" does not, and would not, contain information that would enable one to decode the character "1" or the character "B" back

into an underscore. For these reasons, it is respectfully submitted that Douceur does not teach or suggest an encoder module for encoding a filename by replacing an illegal character with a specific code character having information coded therein about the illegal character itself, as claimed herein.

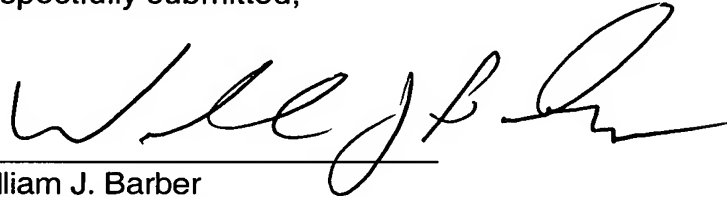
#### Remaining Dependent Claims

The remaining dependent claims depend directly or indirectly from one or more of the aforementioned independent claims, contain all the limitations thereof, and are deemed patentable for all the reasons set forth above.

#### Conclusion

For all these reasons, reconsideration and early allowance is respectfully requested.

Respectfully submitted,

A handwritten signature in black ink, appearing to read 'W. J. Barber', is written over a horizontal line.

William J. Barber  
Attorney for the Applicant  
Registration No. 32,720

15 May 2007  
WARE, FRESSOLA, VAN DER SLUYS  
& ADOLPHSON LLP  
Customer No. 004955  
Bradford Green, Building Five  
755 Main Street, P.O. Box 224  
Monroe, CT 06468  
(203) 261-1234